

# Proof System for Plan Verification under 0-Approximation Semantics

Xishun Zhao <sup>\*†</sup>, Yuping Shen  
Institute of Logic and Cognition,  
Sun Yat-sen University  
510275 Guangzhou, (P.R. China)  
Email: hsszxs@mail.sysu.edu.cn

September 23, 2011

## Abstract

In this paper a proof system is developed for plan verification problems  $\{X\}c\{Y\}$  and  $\{X\}c\{KWp\}$  under 0-approximation semantics for  $\mathcal{A}_K$ . Here, for a plan  $c$ , two sets  $X, Y$  of fluent literals, and a literal  $p$ ,  $\{X\}c\{Y\}$  (resp.  $\{X\}c\{KWp\}$ ) means that all literals of  $Y$  become true (resp.  $p$  becomes known) after executing  $c$  in any initial state in which all literals in  $X$  are true. Then, soundness and completeness are proved. The proof system allows verifying plans and generating plans as well.

**Key words:** Plan Verification; 0-Approximation; Proof System

## 1 Introduction

Planning refers to the procedure of finding a sequence of actions(i.e., a *plan*) which leads a possible world from an initial state to a goal. In the early days

---

<sup>\*</sup>Corresponding author. Tel: 0086-20-84114036, Fax:0086-20-84110298.

<sup>†</sup>This research was partially supported by the NSFC project under grant number: 60970040 and a MOE project under grant number: 05JJD72040122.

of Artificial Intelligence(AI), an agent(i.e., plan generator or executor) was assumed to have complete knowledge about the world but it turned out to be unrealistic. Therefore, planning under *incomplete knowledge* earns a lot of attention since late 1990s [15, 6, 22, 10, 19, 17]. A widely accepted solution is to equip the planner with actions for producing knowledge, also called *sensing actions*, and allow to use *conditional plan*[10, 24, 25, 23, 16], i.e., plans containing *conditional expressions* (e.g., **If-Then-Else** structures).

Consider the following example [24], say a bomb can only be safely defused if its alarm is switched off. Flipping the switch causes the alarm off if it is on and vice versa. At the beginning we only know the bomb is not *disarmed* and not *exploded*, however, we do not know whether or not the alarm is on, i.e., the knowledge about initial state of the domain is incomplete. An agent could correctly defuse the bomb by performing the conditional plan  $c$  below:

$check; \mathbf{If} \text{ alarm\_off } \mathbf{Then} \text{ defuse } \mathbf{Else} \{switch; defuse\}$

in which *check* is a sensing action that produces the knowledge about the alarm. It is necessary to mention that there exists no feasible classical plans for this scenario, e.g., neither *defuse* nor *switch; defuse* could safely disarm the bomb.

To describe and reason about domains with incomplete knowledge, a number of logical frameworks were proposed in the literature. One of well-established formalizations is the *action language*  $\mathcal{A}_K$  [24, 4]. In contrast to its first order antecedents [15, 22],  $\mathcal{A}_K$  possesses a natural syntax and a transition function based semantics, both together provides a flexible mechanism to model the change of an agent's knowledge in a simplified Kripke structure.

In [24] the authors propose several semantics for  $\mathcal{A}_K$ , all of which, roughly speaking, are based on some transition function from pairs of actions and initial states to states. For convenience we use SB-semantics to denote the semantics based on the transition function which maps pairs of actions and c-states to c-states. Here, a c-state is a pair of a world state and a knowledge state which is a set of world states. One of the results in [4] is that the polynomial plan existence problem under SB-semantics is PSPACE-complete. Even we restrict the number of fluents determined by a sensing action, the existence of polynomial plan with limited number sensing actions is  $\Sigma_2^P$ -complete [4]. To overcome the high complexity, Baral and Son [24] have proposed  $i$ -approximations,  $i = 0, 1, \dots$ . It has been proved in [4] that under some restricted conditions polynomial plan existence problem under

0-approximation is NP-complete, that is, it is still intractable because it is widely believed that there is no polynomial algorithm solving an NP-complete problem.

Although modern planners are quite successful to produce and verify short plans they still face a great challenge to generate longer plans. There have been many efforts to construct transformations from planning or plan verification to other logic formalisms, for example, first-order logic (FOL) [11, 9, 24], propositional satisfiability (SAT) [20], QBF satisfiability (QSAT), [18, 14], non-monotonic logics [7, 3, 13], and so on. These approaches provide ways to use existing solvers for planning and plan verification, they do not, however, tell us how to generate and verify new plans from old ones.

It is well known that programming is generally also very hard, however, proof system for program verification allows one to construct new correct programs from shorter ones [1]. Similarly, proof systems for plan verification would be helpful for verifying and constructing longer correct plans.

For a given domain description  $D$ , two sets  $X, Y$  of fluent literals, and a plan  $c$ , we consider the verification problem of determining whether  $D \models \{X\}c\{Y\}$ , that is, whether all literals of  $Y$  becomes true after executing  $c$  in any initial state in which all literals of  $X$  are true. It seems natural that from  $D \models \{X\}c_1\{Y\}$  and  $D \models \{Y\}c_2\{Z\}$  we should obtain  $D \models \{X\}c_1; c_2\{Z\}$ . That is,

$$\frac{\{X\}c_1\{Y\}, \{Y\}c_2\{Z\}}{\{X\}c_1; c_2\{Z\}}$$

should be a valid rule. This paper is devoted to develop a sound and complete proof system for plan verification under 0-approximation.

One important observation is that constructing proof sequences could also be considered as a procedure for generating plans. This feature is very useful for the agent to do so-called *off-line* planning [12, 5]. That is, when the agent is free from assigned tasks, she could continuously compute (short) proofs and store them into a well-maintained database. Such a database consists of a huge number of proofs of the form  $\{X\}c\{Y\}$  after certain amount of time. W.l.o.g., we may assume these proofs are stored into a graph, where  $\{X\}$ ,  $\{Y\}$  are nodes and  $c$  is an connecting edge. With such a database, the agent could do *on-line* query quickly. Precisely speaking, asking whether a plan  $c'$  exists for leading state  $\{X'\}$  to  $\{Y'\}$ , is equivalent to look for a path  $c'$  from  $\{X'\}$  to  $\{Y'\}$  in the graph. This is known as the PATH problem and could be easily computed (NL-complete, see [21]).

The paper is organized as follows. In Section 2 we mainly recall the language of  $\mathcal{A}_K$  and the 0-approximation semantics. In addition, a few new lemmas are proved, which will be used in later sections. Section 3 is devoted to the construction of proof system. Soundness and completeness are proved. Section 4 concludes this paper.

## 2 The Language $\mathcal{A}_K$

The language  $\mathcal{A}_K$  [24] proposed by Baral & Son is a well known framework for reasoning about sensing actions and conditional planning. In this section we recall the syntax and the 0-approximation semantics of  $\mathcal{A}_K$ , in addition we prove several new properties (e.g. the monotonicity of 0-transition function, see Lemma 2.1 below) which will be used in next section.

### 2.1 Syntax of $\mathcal{A}_K$

Two disjoint non-empty sets of symbols, called *fluent names* (or *fluents*) and *action names* (or *actions*) are introduced as the alphabet of the language  $\mathcal{A}_K$ . A *fluent literal* is either a fluent  $f$  or its negation  $\neg f$ . For a fluent  $f$ , by  $\neg\neg f$  we mean  $f$ . For a fluent literal  $p$ , we define  $\text{fln}(p) := f$  if  $p$  is a fluent  $f$  or is  $\neg f$ . Given a set  $X$  of fluent literals,  $\neg X$  is defined as  $\{\neg p \mid p \in X\}$ , and  $\text{fln}(X)$  is defined as  $\{\text{fln}(p) \mid p \in X\}$ .

The language  $\mathcal{A}_K$  uses four kinds *propositions* for describing a domain.

An *initial-knowledge proposition* (which is called v-proposition in [24]) is an expression of the form

$$\text{initially } p \tag{1}$$

where  $p$  is a fluent literal. Roughly speaking, the above proposition says that  $p$  is initially known to be true.

An *effect proposition* (*ef-proposition* for short) is an expression of the form

$$a \text{ causes } p \text{ if } p_1, \dots, p_n \tag{2}$$

where  $a$  is an action and  $p, p_1, \dots, p_n$  are fluent literals. We say  $p$  and  $\{p_1, \dots, p_n\}$  are the *effect* and the *precondition* of the proposition, respectively. The intuitive meaning of the above proposition is that  $p$  is guaranteed to be true after the execution of action  $a$  in any state of the world where  $p_1, \dots, p_n$  are true. If the precondition is empty then we drop the **if** part and simply say:  $a$  **causes**  $p$ .

An *executability proposition* (*ex-proposition* for short) is an expression of the form

$$\textbf{executable } a \textbf{ if } p_1, \dots, p_n \quad (3)$$

where  $a$  is an action and  $p_1, \dots, p_n$  are fluent literals. Intuitively, it says that the action  $a$  is executable whenever  $p_1, \dots, p_n$  are true. For convenience, we call  $\{p_1, \dots, p_n\}$  the *ex-preconditions* of the proposition.

A *knowledge proposition* (*k-proposition* for short) is of the form

$$a \textbf{ determines } f \quad (4)$$

where  $a$  is an action and  $f$  is a fluent. Intuitively, the above proposition says that after  $a$  is executed the agent will know whether  $f$  is true or false.

A *proposition* is either an initial-knowledge proposition, or an ef-proposition, or an ex-proposition, or a k-proposition. Two initial-knowledge propositions **initially**  $f$  and **initially**  $g$  are called *contradictory* if  $f = \neg g$ . Two effect propositions “ $a$  **causes**  $f$  **if**  $p_1, \dots, p_n$ ” and “ $a$  **causes**  $g$  **if**  $q_1, \dots, q_m$ ” are called *contradictory* if  $f = \neg g$  and  $\{p_1, \dots, p_n\} \cap \{\neg q_1, \dots, \neg q_m\}$  is empty.

**Definition 2.1** ([24]) *A domain description in  $\mathcal{A}_K$  is a set of propositions  $D$  which does not contain*

- (1) *contradictory initial-knowledge propositions,*
- (2) *contradictory ef-propositions*

Actions occurring in knowledge propositions are called *sensing actions*, while actions occurring in effect propositions are called *non-sensing actions*. In this paper we request that for any domain description  $D$  the set of sensing actions in  $D$  and the set of non-sensing actions in  $D$  should be disjoint.

**Definition 2.2** (*Conditional Plan* [24]) *A conditional plan is inductively defined as follows:*

1. *The empty sequence of actions, denoted by  $[]$ , is a conditional plan;*
2. *If  $a$  is an action then  $a$  is a conditional plan;*
3. *If  $c_1$  and  $c_2$  are conditional plans then the combination  $c_1; c_2$  is a conditional plan;*

4. If  $c_1, \dots, c_n$  ( $n \geq 1$ ) are conditional plans and  $\varphi_1, \dots, \varphi_n$  are conjunctions of fluent literals (which are mutually exclusive but not necessarily exhaustive) then the following is a conditional plan (also called a case plan):

**case**  $\varphi_1 \rightarrow c_1. \dots . \varphi_n \rightarrow c_n.$  **endcase**

5. Nothing else is a conditional plan.

Propositions are used to describe a domain, whereas *queries* are used to ask questions about the domain. For a plan  $c$ , a set  $X$  of fluent literals, and a fluent literal  $p$ , we have two kinds of queries:

**Knows**  $X$  **after**  $c$  (5)

**Kwhether**  $p$  **after**  $c$  (6)

Intuitively, query of the form (5) asks whether all literals in  $X$  will be known to be true after executing  $c$ , while query of the form (6) asks whether  $p$  will be either known to be true or known to be false after executing  $c$ .

## 2.2 0-Approximation Semantics

In this section we arbitrarily fix a domain description  $D$  without contradictory propositions. From now on when we speak of fluent names and action names we mean that they occur in propositions of  $D$ .

According to [24], an a-state is a pair  $(T, F)$  of two disjoint sets of fluent names. A fluent  $f$  is true (resp. false) in  $(T, F)$  if  $f \in T$  (resp.  $f \in F$ ). Dually,  $\neg f$  is true (resp. false) if  $f$  is false (resp. true). For a fluent name  $f$  outside  $T \cup F$ , both  $f$  and  $\neg f$  are unknown. A fluent literal  $p$  is called possibly true if it is not false (i.e., true or unknown). In the following we often use  $\sigma, \delta$  to denote a-states. For a set  $X = \{p_1, \dots, p_m\}$  of fluent literals, we say  $X$  is true in an a-state  $\sigma$  if and only if every  $p_i$  is true in  $\sigma$ ,  $i = 1, \dots, m$ .

An action  $a$  is said to be 0-executable in an a-state  $\sigma$  if there exists an ex-proposition **executable**  $a$  **if**  $p_1, \dots, p_n$ , such that  $p_1, \dots, p_n$  are true in  $\sigma$ . The following notations were introduced in [24].

- (1)  $e_a^+(\sigma) := \{f \mid f \text{ is a fluent and there exists "a causes } f \text{ if } p_1, \dots, p_n"$   
in  $D$  such that  $p_1, \dots, p_n$  are true in  $\sigma\}$ .
- (2)  $e_a^-(\sigma) := \{f \mid f \text{ is a fluent and there exists "a causes } \neg f \text{ if } p_1, \dots, p_n"$   
in  $D$  such that  $p_1, \dots, p_n$  are true in  $\sigma\}$ .

- (3)  $F_a^+(\sigma) := \{f \mid f \text{ is a fluent and there exists "a causes } f \text{ if } p_1, \dots, p_n"$   
in  $D$  such that  $p_1, \dots, p_n$  possibly true in  $\sigma\}$ .
- (4)  $F_a^-(\sigma) := \{f \mid f \text{ is a fluent and there exists "a causes } \neg f \text{ if } p_1, \dots, p_n"$   
in  $D$  such that  $p_1, \dots, p_n$  are possible true in  $\sigma\}$ .
- (5)  $K(a) := \{f \mid f \text{ is a fluent and "a determines } f"$  is in  $D\}$ .

For an a-sate  $\sigma = (T, F)$  and a non-sensing action  $a$  0-executable in  $\sigma$ , the result after executing  $a$  is defined as

$$\text{Res}_0(a, \sigma) := ((T \cup e_a^+(\sigma)) \setminus F_a^-(\sigma), (F \cup e_a^-(\sigma)) \setminus F_a^+(\sigma))$$

The extension order  $\preceq$  on a-states is defined as follows [24]:

$$(T_1, F_1) \preceq (T_2, F_2) \text{ if and only if } T_1 \subseteq T_2, F_1 \subseteq F_2.$$

Please note that if  $(T_1, F_1) \preceq (T_2, F_2)$  then for a fluent literal  $p$  we have

- if  $p$  is true (resp. false) in  $(T_1, F_1)$  then  $p$  is true (resp. false) in  $(T_2, F_2)$ ,
- if  $p$  is unknown in  $(T_2, F_2)$  then  $p$  must be unknown in  $(T_1, F_1)$ , and
- if  $p$  is possibly true in  $(T_2, F_2)$  then  $p$  is possibly true in  $(T_1, F_1)$ .

Consequently, for any non-sensing action  $a$  and a-states  $\sigma_1$  and  $\sigma_2$  such that  $\sigma_1 \preceq \sigma_2$  and  $a$  is 0-executable in  $\sigma_1$ , we have

- $a$  is 0-executable in  $\sigma_2$ .
- $e_a^+(\sigma_1) \subseteq e_a^+(\sigma_2)$ , and  $e_a^-(\sigma_1) \subseteq e_a^-(\sigma_2)$ .
- $F_a^+(\sigma_2) \subseteq F_a^+(\sigma_1)$ , and  $F_a^-(\sigma_2) \subseteq F_a^-(\sigma_1)$ .

Then we have the following proposition.

**Proposition 2.1** *For any non-sensing action  $a$  and a-states  $\sigma_1$  and  $\sigma_2$  such that  $\sigma_1 \preceq \sigma_2$  and  $a$  is 0-executable in  $\sigma_1$ , we have*

$$\text{Res}_0(a, \sigma_1) \preceq \text{Res}_0(a, \sigma_2).$$

The 0-transition function  $\Phi_0$  of  $D$  is defined as follows [24].

- If  $a$  is not 0-executable in  $\sigma$ , then  $\Phi_0(a, \sigma) := \{\perp\}$ .
- If  $a$  is 0-executable in  $\sigma$  and  $a$  is a non-sensing action,  $\Phi_0(a, \sigma) := \{\text{Res}_0(a, \sigma)\}$ .
- If  $a$  is 0-executable in  $\sigma = (T, F)$  and  $a$  is a sensing action, then  $\Phi_0(a, \sigma) := \{(T', F') \mid (T, F) \preceq (T', F') \text{ and } T' \cup F' = T \cup F \cup K(a)\}$ .
- $\Phi_0(a, \Sigma) := \bigcup_{\sigma \in \Sigma} \Phi_0(a, \sigma)$ .

Let  $\Sigma_1, \Sigma_2$  be two sets of a-states, we write  $\Sigma_1 \preceq \Sigma_2$  if for every a-state  $\delta$  in  $\Sigma_2$ , there is an a-state  $\sigma$  in  $\Sigma_1$  such that  $\sigma \preceq \delta$ .

The next proposition follows directly from Proposition 2.1. and the definition of  $\Phi_0(a, \sigma)$  above.

**Proposition 2.2** *Suppose  $\sigma_1 \preceq \sigma_2$  and  $a$  is an action 0-executable in  $\sigma_1$ , then  $\Phi_0(a, \sigma_1) \preceq \Phi_0(a, \sigma_2)$ .*

The extended 0-transition function  $\widehat{\Phi}_0$ , which maps pairs of conditional plans and a-states into sets of a-states, is defined inductively as follows.

**Definition 2.3** ([24])

$$\widehat{\Phi}_0([], \sigma) := \{\sigma\}$$

$$\widehat{\Phi}_0(a, \sigma) := \Phi_0(a, \sigma)$$

When  $c$  is a case plan **case**  $\varphi_1 \rightarrow c_1 \cdots \varphi_k \rightarrow c_k$ . **endcase**,

$$\widehat{\Phi}_0(c, \sigma) := \begin{cases} \widehat{\Phi}_0(c_j, \sigma), & \text{if } \varphi_j \text{ is true in } \sigma, \\ \{\perp\}, & \text{if non of } \varphi_1, \dots, \varphi_k \text{ is true in } \sigma. \end{cases}$$

$$\widehat{\Phi}_0(c_1; c_2, \sigma) := \bigcup_{\sigma' \in \widehat{\Phi}_0(c_1, \sigma)} \widehat{\Phi}_0(c_2, \sigma')$$

$$\widehat{\Phi}_0(c, \perp) := \{\perp\}.$$

$$\widehat{\Phi}_0(c, \Sigma) := \bigcup_{\sigma \in \Sigma} \widehat{\Phi}_0(c, \sigma).$$

**Remark 2.1** From the definitions above we know that transition functions  $\Phi_0$  and  $\widehat{\Phi}_0$  of a domain description  $D$  do not depends on any initial-knowledge proposition. In other words, if two domain descriptions  $D_1$  and  $D_2$  contain the same non initial-knowledge propositions, then their transition functions coincide.



A condition plan  $c$  is 0-executable in  $\sigma$  if  $\perp \notin \widehat{\Phi}_0(c, \sigma)$ .

**Lemma 2.1** (*Monotonicity Lemma*) *Let  $c$  be a plan,  $\Sigma_1, \Sigma_2$  be two sets of a-states. Suppose  $\Sigma_1 \preceq \Sigma_2$ , and  $c$  is 0-executable in every a-state on  $\Sigma_1$ . Then  $\widehat{\Phi}_0(c, \Sigma_1) \preceq \widehat{\Phi}_0(c, \Sigma_2)$ .*

**Proof:** We proceed by induction on the structure of the plan  $c$ .

1. Suppose  $c$  consists of only an action  $a$ . Consider an arbitrary a-state  $\sigma'_2 \in \Phi_0(a, \Sigma_2)$ . Then there is an a-state  $\sigma_2 = (T_2, F_2) \in \Sigma_2$  such that  $\sigma'_2 \in \Phi_0(a, \sigma_2)$ . Since  $\Sigma_1 \preceq \Sigma_2$ , pick  $\sigma_1 = (T_1, F_1) \in \Sigma_1$  such that  $\sigma_1 \preceq \sigma_2$ . It is sufficient to show that  $\sigma'_1 \preceq \sigma'_2$  for some  $\sigma'_1 \in \Phi_0(a, \Sigma_1)$ .

If  $a$  is a non-sensing action  $a$ , then the assertion follows directly from Proposition 2.2. Suppose  $a$  is a sensing action. Then  $\sigma'_2$  must be of the form  $(T_2 \cup X, F_2 \cup Y)$  because  $a$  is a sensing action, here  $X \cup Y = K(a)$ . Then clearly  $(T_1 \cup X, F_1 \cup Y)$  must be in  $\Phi_0(a, \Sigma_1)$ . The assertion follows since  $(T_1 \cup X, F_1 \cup Y) \preceq (T_2 \cup X, F_2 \cup Y)$ .

2. Suppose  $c$  is case plan **case**  $\varphi_1 \rightarrow c_1 \cdots \varphi_k \rightarrow c_k$ . **endcase**. Consider any a-state  $\sigma'_2 \in \widehat{\Phi}_0(c, \Sigma_2)$ . Let  $\sigma_1 \in \Sigma_1, \sigma_2 \in \Sigma_2$  be such that  $\sigma_1 \preceq \sigma_2$  and  $\sigma'_2 \in \widehat{\Phi}_0(c, \sigma_2)$ . Since  $c$  is 0-executable in  $\sigma_1$ , some  $\varphi_i$  is true in  $\sigma_1$ . Then  $\varphi_i$  is also true in  $\sigma_2$  since  $\sigma_1 \preceq \sigma_2$ . Then by the induction hypothesis,  $\widehat{\Phi}_0(c, \sigma_1) = \widehat{\Phi}_0(c_i, \sigma_1) \preceq \widehat{\Phi}_0(c_i, \sigma_2) = \widehat{\Phi}_0(c, \sigma_2)$ . Thus, there is  $\sigma'_1 \in \widehat{\Phi}_0(c, \Sigma_1)$  such that  $\sigma'_1 \preceq \sigma'_2$ . Consequently,  $\widehat{\Phi}_0(c, \Sigma_1) \preceq \widehat{\Phi}_0(c, \Sigma_2)$ .
3. Suppose  $c = c_1; c_2$ . By induction hypothesis  $\widehat{\Phi}_0(c_1, \Sigma_1) \preceq \widehat{\Phi}_0(c_1, \Sigma_2)$ . Then by the definition of  $\widehat{\Phi}_0$  we have

$$\widehat{\Phi}_0(c, \Sigma_1) = \left( \bigcup_{\sigma' \in \widehat{\Phi}_0(c_1, \Sigma_1)} \widehat{\Phi}_0(c_2, \sigma') \right) \preceq \left( \bigcup_{\sigma'' \in \widehat{\Phi}_0(c_1, \Sigma_2)} \widehat{\Phi}_0(c_2, \sigma'') \right) = \widehat{\Phi}_0(c, \Sigma_2)$$

■

An a-state  $\sigma$  is called an initial a-state of  $D$  if  $p$  is true in  $\sigma$  for any fluent literal  $p$  such that the initial-knowledge proposition “**initially**  $p$ ” is in  $D$ .

Suppose  $D$  is a domain description,  $c$  is a conditional plan,  $X$  is a set of fluent literals, and  $p$  a literals. The semantics for the queries are given below:

**Definition 2.4** ([24])

- $D \models_0 \mathbf{Knows} X \text{ after } c$  if for every initial a-state  $\sigma$ , the plan  $c$  is 0-executable in  $\sigma$ , and  $X$  is true in every a-state in  $\widehat{\Phi}_0(c, \sigma)$ .
- $D \models_0 \mathbf{Kwhether} p \text{ after } c$  if for every initial a-state  $\sigma$ , the plan  $c$  is 0-executable in  $\sigma$ , and  $p$  is either true or false in every a-state in  $\widehat{\Phi}_0(c, \sigma)$ .

Let  $T_D := \{f \mid \text{"initially } f" \in D\}$ ,  $F_D := \{f \mid \text{"initially } \neg f" \in D\}$ . Obviously,  $(T_D, F_D)$  is the least initial a-state of  $D$ , that is,  $(T_D, F_D) \preceq \sigma$  for any initial a-state  $\sigma$ . The following lemma follows easily from Lemma 2.1.

**Lemma 2.2**

- $D \models_0 \mathbf{Knows} X \text{ after } c$  if and only if the plan  $c$  is 0-executable in  $(T_D, F_D)$ , and  $X$  true in every a-state in  $\widehat{\Phi}(c, (T_D, F_D))$ .
- $D \models_0 \mathbf{Kwhether} p \text{ after } c$  if the plan  $c$  is 0-executable in  $(T_D, F_D)$ , and  $p$  is either true or false in every a-state in  $\widehat{\Phi}(c, (T_D, F_D))$ .

### 3 A Proof System for 0-Approximation

A consistent set  $X$  of literals determines a unique a-state  $(T_X, F_X)$  by  $T_X := \{f \mid f \in X\}$  and  $F_X := \{f \mid \neg f \in X\}$ . And conversely an a-state determines uniquely the set  $S_{(T,F)} := T \cup \neg F$ . Obviously,  $p \in X$  if and only if  $p$  is true in  $(T_X, F_X)$  for any literal  $p$ .

In the following we will not distinguish sets of literals and a-states from each other. For example,  $\text{Res}_0(a, X)$  is nothing but  $\text{Res}_0(a, (T_X, F_X))$  which can be regarded as a set of literals. Analogically, we have notations  $\Phi_0(c, X)$  and  $\widehat{\Phi}_0(c, X)$ , which can be regarded as collections of sets of literals.

**Definition 3.1** Let  $D$  be a domain description without initial-knowledge propositions. Suppose  $X, Y$  are two sets of fluent literals. By  $D \models_0 \{X\}c\{Y\}$  we mean  $D \cup \text{ini}(X) \models_0 \mathbf{Knows} Y \text{ after } c$ . Here  $\text{ini}(X) = \{\text{initially } p \mid p \in X\}$ .

**Remark 3.1**

- The idea of the notation  $\{X\}c\{Y\}$  comes from programming verification where in the sense of total correctness  $\{\varphi\}P\{\psi\}$  means that any computation of  $P$  starts in a state satisfying  $\varphi$  will terminates in a state satisfying  $\psi$ . (see e.g. [1])
- By Lemma 2.2,  $D \models_0 \{X\}c\{Y\}$  if and only if  $Y$  is true in every a-state in  $\widehat{\Phi}_0(c, X)$ .

Suppose  $D$  is a general domain description (that is, initially-knowledge propositions are allowed). Let  $D'$  be the set of all non-initial-knowledge propositions of  $D$ , and let  $X := \{p \mid \text{“initially } p\text{” is in } D\}$ . Then  $D' \models_0 \{X\}c\{Y\}$  is equivalent to  $D \models_0 \mathbf{Knows } Y \text{ after } c$ .

### 3.1 The Proof System $\text{PR}_D^0$ for Knows

In the remainder of this section we fixed a domain description  $D$  without initial-knowledge propositions. We always use  $X, Y, X', Y'$  to denote consistent set of fluent literals. The proof system  $\text{PR}_D^0$  consists of the following groups of axioms and rules 1-6.

AXIOM 1. (Empty)

$$\{X\}[]\{X\}.$$

AXIOM 2. (Non-sensing Action)

$$\{X\}a\{(\text{Res}_0(a, X))\}.$$

Where  $a$  is a non-sensing action 0-executable in  $X$ .

RULE 3. (Sensing Action)

$$\frac{\{X \cup X_1\}c\{Y\}, \dots, \{X \cup X_m\}c\{Y\}}{\{X\}a; c\{Y\}}.$$

Where  $a$  is a sensing action 0-executable in  $X$ , and  $X_1, \dots, X_m$  are all sets  $X'$  of fluent literals such that  $\text{fln}(X') = K(a)$  and  $X \cup X'$  is consistent.

RULE 4. (Case)

$$\frac{\varphi_i \subseteq X, \{X\}c_i; c'\{Y\}}{\{X\}c; c'\{Y\}}.$$

Where  $c$  is the case plan **case**  $\varphi_1 \rightarrow c_1. \dots. \varphi_m \rightarrow c_m$ . **endcase**.

RULE 5. (Composition)

$$\frac{\{X\}c_1\{Y'\}, \{Y'\}c_2\{Y\}}{\{X\}c_1; c_2\{Y\}}.$$

RULE 6 (Consequence)

$$\frac{X' \subseteq X, \{X'\}c\{Y'\}, Y \subseteq Y'}{\{X\}c\{Y\}}.$$

**Definition 3.2** A proof sequence (or, derivation) of  $PR_D^0$  is a sequence  $\{X_1\}c_1\{Y_1\}, \dots, \{X_n\}c_n\{Y_n\}$  such that each  $\{X_i\}c_i\{Y_i\}$  is either an axiom in  $PR_D^0$  or is obtained from some of  $\{X_1\}c_1\{Y_1\}, \dots, \{X_{i-1}\}c_{i-1}\{Y_{i-1}\}$  by applying a rule in  $PR_D^0$ .

By  $D \vdash_0 \{X\}c\{Y\}$ , we mean that  $\{X\}c\{Y\}$  appears in some proof sequence of  $PR_D^0$ , that is,  $\{X\}c\{Y\}$  can be derived from axioms and rules in  $PR_D^0$ .

**Example 3.1** ([24]) Let

$$D := \left\{ \begin{array}{l} \text{check } \mathbf{determines} \text{ alarm\_off} \\ \text{defuse } \mathbf{causes} \text{ disarmed if alarm\_off} \\ \text{defuse } \mathbf{causes} \text{ exploded if } \neg \text{alarm\_off} \\ \text{switch } \mathbf{causes} \neg \text{alarm\_off if alarm\_off} \\ \text{switch } \mathbf{causes} \text{ alarm\_off if } \neg \text{alarm\_off} \\ \mathbf{executable} \text{ check if } \neg \text{exploded} \\ \mathbf{executable} \text{ switch if } \neg \text{exploded} \\ \mathbf{executable} \text{ defuse if } \neg \text{exploded} \end{array} \right\}$$

Let  $c'$  be the case plan: **case**  $\neg \text{alarm\_off} \rightarrow \text{switch}$ .  $\text{alarm\_off} \rightarrow []$ . **endcase**, and  $c$  be the plan:  $\text{check}; c'; \text{defuse}$ . Then the following is a proof sequence of  $PR_D^0$ .

- (1)  $\{\neg \text{disarmed}, \neg \text{exploded}, \neg \text{alarm\_off}\} \text{switch} \{\neg \text{disarmed}, \neg \text{exploded}, \text{alarm\_off}\}$   
(AXIOM 2)
- (2)  $\{\neg \text{disarmed}, \neg \text{exploded}, \neg \text{alarm\_off}\} c' \{\neg \text{disarmed}, \neg \text{exploded}, \text{alarm\_off}\}$   
((1) and RULE 4)
- (3)  $\{\neg \text{disarmed}, \neg \text{exploded}, \text{alarm\_off}\} [] \{\neg \text{disarmed}, \neg \text{exploded}, \text{alarm\_off}\}$   
(AXIOM 1)

- (4)  $\{\neg disarmed, \neg exploded, alarm\_off\}c'\{\neg disarmed, \neg exploded, alarm\_off\}$   
((3) and RULE 4)
- (5)  $\{\neg disarmed, \neg exploded\}check; c'\{\neg disarmed, \neg exploded, alarm\_off\}$   
((2), (4) and RULE 3)
- (6)  $\{\neg disarmed, \neg exploded, alarm\_off\}defuse\{disarmed, \neg exploded, alarm\_off\}$   
(AXIOM 2)
- (7)  $\{\neg disarmed, \neg exploded\}c\{disarmed, \neg exploded, alarm\_off\}$   
((6) and RULE 5)

**Remark 3.2** One important observation is that constructing a proof sequence could also be considered as a procedure for generating plans. This feature is very useful for the agent to do so-called *off-line* planning [12, 5]. That is, when the agent is free from assigned tasks, she could continuously compute (short) proofs and store them into a well-maintained database. Such a database consists of a huge number of proofs of the form  $\{X\}c\{Y\}$  after certain amount of time. W.l.o.g., we may assume these proofs are stored into a graph, where  $\{X\}$ ,  $\{Y\}$  are nodes and  $c$  is an connecting edge. With such a database, the agent could do *on-line* query quickly. Precisely speaking, asking whether a plan  $c'$  exists for leading state  $\{X'\}$  to  $\{Y'\}$ , is equivalent to look for a path  $c'$  from  $\{X'\}$  to  $\{Y'\}$  in the graph. This is known as the PATH problem and could be easily computed (NL-complete, see [21]).

### 3.1.1 Soundness of $PR_D^0$

**Theorem 3.1** (Soundness of  $PR_D^0$ )  *$PR_D^0$  is sound. That is, for any conditional plan  $c$  and any consistent sets  $X, Y$  of fluent literals,  $D \vdash_0 \{X\}c\{Y\}$  implies  $D \models_0 \{X\}c\{Y\}$ .*

**Proof:** Suppose  $D \vdash_0 \{X\}c\{Y\}$ . Then  $\{X\}c\{Y\}$  has a derivation. We shall proceed by induction on the length of the derivation. Let  $\Phi_0$  and  $\hat{\Phi}_0$  be 0-transition functions of  $D$ . Please note that for any set  $S$  of fluent literals, the 0-transition functions of  $D \cup \text{ini}(S)$  are the same as  $\Phi_0$  and  $\hat{\Phi}_0$ , respectively (see Remark 2.1).

1. Suppose  $\{X\}c\{Y\}$  is an axiom in AXIOM 1. Then  $X = Y$  and  $c = []$ . Clearly,  $D \models_0 \{X\}[]\{X\}$ .

2. Suppose  $\{X\}c\{Y\}$  is an axiom in AXIOM 2, i.e.,  $c$  consists of only a non-sensing action  $a$  which is 0-executable in  $X$ , and  $Y = \text{Res}_0(a, X)$ . Since  $\widehat{\Phi}_0(a, X) = \{\text{Res}_0(a, X)\}$ , it follows that  $D \models_0 \{X\}a\{Y\}$ .
3. Suppose  $\{X\}c\{Y\}$  is obtained by applying a rule in RULE 3. Then  $c = a; c_1$  for some sensing action  $a$  0-executable in  $X$ , and  $\{X\}c\{Y\}$  is obtained from  $\{X \cup X_1\}c_1\{Y\}, \dots, \{X \cup X_m\}c_1\{Y\}$ , where  $X_1, \dots, X_m$  are all sets  $X'$  of fluent literals such that  $\text{fln}(X') = K(a)$  and  $X \cup X'$  is consistent. By the induction hypothesis,

$$D \models_0 \{X \cup X_i\}c_1\{Y\}, \text{ for } i = 1, \dots, m.$$

That is, all literals in  $Y$  are true in every set in  $\widehat{\Phi}_0(c_1, X \cup X_i)$ . Please note that  $\Phi_0(a, X) = \{X \cup X_1, \dots, X \cup X_m\}$ . By the definition of  $\widehat{\Phi}_0$  (see Definition 2.3),

$$\widehat{\Phi}_0(c, X) = \bigcup_{i=1}^m \widehat{\Phi}_0(c_1, X \cup X'_i).$$

Therefore,  $D \models_0 \{X\}c\{Y\}$ .

4. Suppose  $\{X\}c\{Y\}$  is obtained by applying a rule in RULE 4. That is,  $c$  is a plan  $c_1; c_2$ , where  $c_1$  is a case plan **case**  $\varphi_1 \rightarrow c'_1. \dots \varphi_n \rightarrow c'_n$ . **endcase** such that for some  $i \in \{1, \dots, n\}$ ,  $\varphi_i \subseteq X$  and  $\{X\}c'_i; c_2\{Y\}$  has been derived. By the induction hypothesis, we have  $D \models_0 \{X\}c'_i; c_2\{Y\}$ . By Definition 2.3, we have  $\widehat{\Phi}_0(c, X) = \widehat{\Phi}_0(c_2, \widehat{\Phi}_0(c_1, \sigma)) = \widehat{\Phi}_0(c_2, \widehat{\Phi}_0(c'_i, X)) = \widehat{\Phi}_0(c'_i; c_2, X)$ . Then, all literals of  $Y$  are true in  $\widehat{\Phi}_0(c, X)$ . Thus,  $D \models_0 \{X\}c\{Y\}$ .
5. Suppose  $\{X\}c\{Y\}$  is obtained from  $\{X\}c_1\{Y'\}$  and  $\{Y'\}c_2\{Y\}$  by applying a rule in RULE 5. By the inductive hypothesis,

$$D \models_0 \{X\}c_1\{Y'\} \text{ and } D \models_0 \{Y'\}c_2\{Y\}.$$

Then for any  $S \in \widehat{\Phi}_0(c_1, X)$ , we have  $Y' \subseteq S$  (i.e.,  $(T_{Y'}, F_{Y'}) \preceq (T_S, F_S)$ ). Thus, by Lemma 2.1,  $\widehat{\Phi}_0(c_2, Y') \preceq \widehat{\Phi}_0(c_2, S)$ . Then

$$\widehat{\Phi}_0(c_2, Y') \preceq \left( \bigcup_{S \in \widehat{\Phi}_0(c_1, X)} \widehat{\Phi}_0(c_2, S) \right) = \widehat{\Phi}_0(c, X),$$

It follows that  $D \models_0 \{X\}c\{Y\}$ .

6. Suppose  $\{X\}c\{Y\}$  is obtained by applying a rule in RULE 6. That is, there is  $X' \subseteq X$  and  $Y' \supseteq Y$  such that  $\{X'\}c\{Y'\}$  has been derived. Then by the induction hypothesis, all literals in  $Y'$  is known to be true in  $\widehat{\Phi}_0(c, X')$ , so are literals in  $Y$ . By Lemma 2.1 we have  $\widehat{\Phi}_0(c, X') \preceq \widehat{\Phi}_0(c, X)$ . Therefore,  $D \models_0 \{X\}c\{Y\}$ .

Altogether, we complete the proof. ■

### 3.1.2 Completeness of $\mathbf{PR}_D^0$

**Theorem 3.2** (*Completeness of  $\mathbf{PR}_D^0$* )  $\mathbf{PR}_D^0$  is complete. That is, for any conditional plan  $c$  and any consistent sets  $X, Y$  of fluent literals,  $D \models_0 \{X\}c\{Y\}$  implies  $D \vdash_0 \{X\}c\{Y\}$ .

**Proof:** Suppose  $D \models_0 \{X\}c\{Y\}$ . We shall show  $D \vdash_0 \{X\}c\{Y\}$ . We shall proceed by induction on the structure of  $c$ .

1. Suppose  $c$  consists of only an action  $a$ . Then  $a$  is 0-executable in  $X$ .
  - **Case 1.**  $a$  is a non-sensing action. Then all literals in  $Y$  are true in  $\text{Res}_0(a, X)$ , that is,  $Y \subseteq \text{Res}_0(a, X)$ . By Axiom 2,  $D \vdash_0 \{X\}a\{\text{Res}_0(a, X)\}$ . Then by RULE 6, we obtain  $D \vdash_0 \{X\}a\{Y\}$ .
  - **Case 2.**  $a$  is a sensing action. Consider any  $p \in Y$ . We shall show  $p \in X$ . Suppose otherwise, then  $X' := X \cup \{\neg p\}$  is still consistent. Then  $\Phi_0(a, X) \preceq \Phi_0(a, X')$ . Thus  $p$  should also be true in every a-state in  $\Phi_0(a, X')$ . On the other hand,  $\neg p$  is true in every a-state in  $\Phi_0(a, X')$  since  $\neg p \in X'$ . This is a contradiction. Thus  $Y \subseteq X$ . Then for any set  $X'$  such that  $\text{fln}(X') = K(a)$  and  $X \cup X'$  is consistent, we have  $D \vdash_0 \{X \cup X'\}[]\{Y\}$ . Now applying RULE 3 we obtain  $D \vdash_0 \{X\}a\{Y\}$ .
2. Suppose  $c$  is a case plan **case**  $\varphi_1 \rightarrow c_1. \dots \varphi_m \rightarrow c_m$ . **endcase**. Since  $D \models_0 \{X\}c\{Y\}$ , it follows that  $\varphi_i \subseteq X$  for some  $i$  (otherwise,  $c$  would not be 0-executable in  $X$ ). Then  $D \models_0 \{X\}c_i\{Y\}$ . By the induction hypothesis,  $D \vdash_0 \{X\}c_i\{Y\}$ . By RULE 4 we have  $D \vdash_0 \{X\}c\{Y\}$ .
3. Suppose  $c$  is a composition plan  $c_1; c_2$ . We shall show the assertion by induction on the structure of  $c_1$ .

- $c_1$  is a non-sensing action  $a$ . By Definition 2.3,  $\widehat{\Phi}_0(a; c_2, X) = \widehat{\Phi}_0(c_2, \text{Res}_0(a, X))$ . By the induction hypothesis,  $D \vdash_0 \{\text{Res}_0(a, X)\}_{c_2}\{Y\}$ . By AXIOM 2 and RULE 5, we obtain  $D \vdash_0 \{X\}_{c_1}\{Y\}$ .
- $c_1$  is a sensing action  $a$ . Consider any  $X'$  such that  $\text{fln}(X') = K(a)$  and  $X \cup X'$  is consistent. Since  $D \models_0 \{X\}a; c_2\{Y\}$ , it follows  $D \models_0 \{X \cup X'\}_{c_2}\{Y\}$ . Then by the induction hypothesis we have  $D \vdash_0 \{X \cup X'\}_{c_2}\{Y\}$ . By RULE 3 we obtain  $D \vdash_0 \{X\}a; c_2\{Y\}$ .
- $c$  is a case plan **case**  $\varphi_1 \rightarrow c'_1. \dots \varphi_m \rightarrow c'_m$ . **endcase**. Since  $c$  is 0-executable in  $X$ , it follows that  $\varphi_i \subseteq X$  for some  $i$ . Then  $D \models_0 \{X\}c'_i; c_2\{Y\}$ . By the induction hypothesis,  $D \vdash_0 \{X\}c'_i; c_2\{Y\}$ . By RULE 4 we have  $D \vdash_0 \{X\}c_1; c_2\{Y\}$ .
- $c_1$  is  $c'_1; c''_1$  such that  $c'$  and  $c''$  are not empty. Then  $c$  is  $c'_1; (c''_1; c_2)$ . Now  $c'_1$  is shorter. By the induction hypothesis,  $D \vdash_0 \{X\}c\{Y\}$ .

Altogether, we complete the proof. ■

### 3.2 The Proof System $\text{PRKW}_D^0$ for Knows-Whether

In this section we shall construct a proof system for reasoning about **Kwhether**  $p$  **after**  $c$  (here  $p$  is a fluent literal). We also fix an arbitrary domain description  $D$  without initial knowledge-propositions. Similar to the notation  $\{X\}c\{Y\}$ , we introduce notation  $\{X\}c\{\text{KW}p\}$ .

**Definition 3.3** *Let  $c$  be a plan,  $X$  be a consistent set of fluent literals, and  $p$  a fluent literal. By  $D \models_0 \{X\}c\{\text{KW}p\}$  we mean*

$$D \cup \text{ini}(X) \models_0 \mathbf{Kwhether} \ p \ \mathbf{after} \ c.$$

Proof system  $\text{PRKW}_D^0$  consists of axioms and rules of groups 1-6 in Section 3.1 and the following groups 7-12.

AXIOM 7.

$$\{X\}a\{\text{KW}f\}$$

Where  $a$  is a sensing action 0-executable in  $X$ , and  $f$  is a fluent name such that the k-proposition “ $a$  **determines**  $f$ ” belongs to  $D$ .



RULE 8.

$$\frac{\{X\}c\{\{p\}\}}{\{X\}c\{KWp\}}$$

RULE 9.

$$\frac{\{X\}c\{KWp\}}{\{X\}c\{KW\neg p\}}$$

RULE 10. (Sensing Action)

$$\frac{\{X \cup X_1\}c\{KWp\}, \dots, \{X \cup X_m\}c\{KWp\}}{\{X\}a; c\{KWp\}}.$$

Where  $a$  is a sensing action 0-executable in  $X$ , and  $X_1, \dots, X_m$  are all sets  $X'$  of fluent literals such that  $\text{fln}(X') = K(a)$  and  $X \cup X'$  is consistent.

RULE 11. (Composition)

$$\frac{\{X\}c_1\{Y\}, \{Y\}c_2\{KWp\}}{\{X\}c_1; c_2\{KWp\}}$$

RULE 12. (Case)

$$\frac{\varphi_i \subseteq X, \{X\}c_i; c'\{KWp\}}{\{X\}c; c'\{KWp\}}.$$

Where  $c$  is the case plan **case**  $\varphi_1 \rightarrow c_1 \dots \varphi_n \rightarrow c_n$ . **endcase**.

**Definition 3.4 (Proof Sequence of  $PRKW_D^0$ )** A Proof sequence (or, derivation) of  $PRKW_D^0$  is a sequence of elements with the form  $\{S_1\}c_1\{T\}$  or  $\{S\}c\{KWp\}$  such that each element is either an axiom in  $PRKW_D^0$  or is obtained from some of previous elements by applying a rule in  $PRKW_D^0$ .

By  $D \vdash_0 \{S\}c\{KWp\}$ , we mean that  $\{S\}c\{KWp\}$  appears in some proof sequence of  $PRKW_D^0$ , that is  $\{S\}c\{KWp\}$  can be derived from axioms and rules in  $PRKW_D^0$ .

**Remark 3.3** Please note that  $\{X\}c\{KWp\}$  never appears as a premise in a rule with consequence of the form  $\{X'\}c'\{Y'\}$ . Thus,  $\{X\}c\{Y\}$  is derivable in  $PRKW_D^0$  if and only if it is derivable in  $PR_D^0$ . So, for derivability of  $\{X\}c\{Y\}$  in  $PRKW_D^0$ , we still employ the notation  $D \vdash_0 \{X\}c\{Y\}$ .

**Theorem 3.3** (*soundness of  $PRKW_D^0$* ) Given a plan  $c$ , then  $D \vdash_0 \{X\}c\{KWp\}$  implies  $D \models_0 \{X\}c\{KWp\}$  for any consistent set  $X$  of fluent literals, and any fluent literal  $p$ .

**Proof:** We can show this theorem by induction on the length of derivations. By the soundness of  $PR_D^0$ , there are six cases according to whether  $\{S\}c\{KWp\}$  is an axiom in AXIOM 7 or obtained by applying a rule in group 8-12. For each case, the proof is easy. We omit the proof. ■

**Theorem 3.4** (*completeness of  $PRKW_D^0$* ) Given a plan  $c$ , then  $D \models_0 \{X\}c\{KWp\}$  implies  $D \vdash_0 \{X\}c\{KWp\}$  for any consistent set  $X$  of fluent literals, and any fluent literal  $p$ .

**Proof:** We proceed by induction on the structure of  $c$ . Suppose  $D \models_0 \{X\}c\{KWp\}$ .

1.  $c$  is empty. Then it must be that  $p \in X$  or  $\neg p \in X$ . Then  $\{X\}[]\{\{p\}\}$  or  $\{X\}[]\{\{\neg p\}\}$  is derivable. Then by RULE 8-9 we can derive  $\{X\}[]\{KWp\}$ .
2.  $c$  consists of only a sensing action  $a$ . Then  $a$  is 0-executable in  $X$ . If  $p \in X$ , it is clearly that  $\{X\}a\{\{p\}\}$  is derivable. From RULE 8 we derive  $\{X\}a\{KWp\}$ . By the same argument, if  $\neg p \in X$ , then  $D \vdash_0 \{X\}a\{KW\neg p\}$ , and then we can derive  $\{X\}a\{KWp\}$  by applying RULE 9. Now we suppose neither  $p$  nor  $\neg p$  is in  $X$ . We claim that the k-proposition “ $a$  **determines**  $\text{fln}(p)$ ” belongs to  $D$  (Otherwise,  $p$  and  $\neg p$  would remain unknown in every a-state in  $\Phi_0(a, X)$ . This contradicts the assumption  $D \models_0 \{X\}a\{KWp\}$ ). Now we have an axiom  $\{X\}a\{KW \text{ fln}(p)\}$ . If  $p$  itself is a fluent name then we are down, else we derive  $\{X\}c\{KWp\}$  by applying RULE 9.
3.  $c$  consists of only a non-sensing action  $a$ . Since  $D \models_0 \{X\}a\{KWp\}$ , it follows that  $a$  is 0-executable in  $X$  and either  $p$  or  $\neg p$  is true in  $\text{Res}_0(a, X)$ . That is,  $p \in \text{Res}(a, X)$  or  $\neg p \in \text{Res}(a, X)$ . Since  $D \vdash_0 \{X\}a\{\text{Res}(a, X)\}$ , we have  $D \vdash_0 \{X\}a\{\{p\}\}$  or  $D \vdash_0 \{X\}a\{\{\neg p\}\}$ . Then either  $\{X\}a\{KWp\}$  or  $\{X\}a\{KW\neg p\}$  can be derived by applying RULE 8. If  $\{X\}a\{KW\neg p\}$  is derivable then we obtain  $\{X\}a\{KWp\}$  by applying RULE 9.
4.  $c$  is a case plan of the form **case**  $\varphi_1 \rightarrow c_1. \dots \varphi_n \rightarrow c_n$ . **endcase**. Then there must be some  $i \in \{1, \dots, n\}$  such that  $\varphi_i \subseteq X$ . Otherwise,  $c$  would not be 0-executable. Then we can see that  $D \models_0 \{X\}c_i\{KWp\}$ .

By the induction hypothesis, we have  $D \vdash_0 \{X\}c_i\{\text{KW}p\}$ . Then we can derive  $\{X\}c\{\text{KW}p\}$  by RULE 12.

5. Suppose  $c = c_1; c_2$  such that  $c_1$  and  $c_2$  are non-empty. We show  $D \vdash_0 \{X\}c\{\text{KW}p\}$  by induction on the structure of  $c_1$ .

- $c_1$  is a sensing action  $a$ . Let  $X_1, \dots, X_m$  be all sets  $X'$  of fluent literals such that  $\text{fln}(X') = K(a)$  and  $X \cup X'$  is consistent. Consider an arbitrary  $X_i$ . We have  $D \models_0 \{X \cup X_i\}c_2\{\text{KW}p\}$  since  $\widehat{\Phi}_0(c_2, X \cup X_i) \subseteq \widehat{\Phi}_0(a; c_2, X)$ . By the induction hypothesis,  $D \vdash_0 \{X \cup X_i\}c_2\{\text{KW}p\}$ . Now by RULE 10 we can derive  $\{X\}a; c_2\{\text{KW}p\}$ .
- $c_1$  is a non-sensing action  $a$ . Then  $a$  is 0-executable in  $X$ . Since  $\widehat{\Phi}_0(c_2, \text{Res}_0(a, X)) = \widehat{\Phi}_0(a; c_2, X)$ , it follows that  $D \models_0 \{\text{Res}_0(a, X)\}c_2\{\text{KW}p\}$ . By the induction hypothesis,  $\{\text{Res}(a, X)\}c_2\{\text{KW}p\}$  is derivable. Please note that  $\{X\}a\{\text{Res}(a, X)\}$  is an axiom in AXIOM 2. By RULE 11, we can derive  $\{X\}a; c_2\{\text{KW}p\}$ .
- $c_1$  is a case plan **case**  $\varphi_1 \rightarrow c'_1. \dots. \varphi_n \rightarrow c'_n$ . **endcase**. We know that  $\varphi_i \subseteq X$  for some  $i \in \{1, \dots, n\}$ . It follows that  $D \models_0 \{X\}c'_i; c_2\{\text{KW}p\}$  since we have assumed  $D \models_0 \{X\}c_1; c_2\{\text{KW}p\}$ . By the induction hypothesis,  $D \vdash_0 \{S\}c'_i; c_2\{\text{KW}p\}$ . Now applying RULE 12 we can derive  $\{X\}c_1; c_2\{\text{KW}p\}$ .
- $c_1 = c'_1; c'_2$  such that  $c'_1, c'_2$  are not empty plan. Then  $c = c'_1; (c'_2; c_2)$ . Now  $c'_1$  is shorter. Then  $\{X\}c\{\text{KW}p\}$  is derivable by the induction hypothesis.

Altogether, we complete the proof. ■

## 4 Conclusions

In this paper, we have proposed a proof system for plan verification under 0-approximation semantics introduced in [24]. The proof system has the following advantages: it is self-contained, hence it does not rely on any particular logic, and need not to pay extra costs to the process of translation; it could be used for both plan verification or plan generation. Particularly, we would like to point out that proof system based inference approach possesses a very desirable property for *off-line* planning. Simply speaking, it allows

the agent to produce and store (shorter) proofs into a database in spare time, and perform quick *on-line* planning by constructing requested proofs from the (shorter) proofs in the database.

Please note that the construction of the proof systems  $\text{PRKW}_D^0$  depends essentially on the monotonicity property of  $\Phi_0$  (see Lemma 2.1). According to [24], an action  $a$  is 1-executable in an a-state  $\sigma$  if it is 0-executable in every complete a-state extending  $\sigma$ . And if a non-sensing action  $a$  is 1-executable in  $\sigma$ , then  $\text{Res}_1(a, \sigma)$  is defined as the intersection of all  $\text{Res}_0(a, \sigma')$ ,  $\sigma' \in \text{Comp}(\sigma)$  which is the set of all complete a-states extending  $\sigma$ . Obviously,  $\text{Res}_1$  is monotonic, that is, if  $\sigma \preceq \delta$  then  $\text{Res}_1(a, \sigma) \preceq \text{Res}_1(a, \delta)$ . Thus the transition function  $\Phi_1$  and  $\widehat{\Phi}_1$  (for precise definition please see [24]) are also monotonic. Therefore, in  $\text{PRKW}_D^0$ , if we replace  $\{X\}a\{\text{Res}_0(a, X)\}$  in AXIOM 2 by  $\{X\}a\{\text{Res}_1(a, X)\}$ , and replace in all groups “0-executable” by “1-executable”, we will obtain a sound and complete proof system  $\text{PRKW}_D^1$  for plan verification under 1-approximation. Please note, however, since 1-executability is unlikely solvable in poly-time, to determine whether a rule in  $\text{PRKW}_D^1$  is applicable seems intractable.

The work of Matteo Baldoni *et al* [2] is closely related to our idea. They proposed a modal logic approach for reasoning about sensing actions, together with goal directed proof procedure for generating conditional plans. The states of a world are represented in [2] as three valued models, so queries about Knows-Whether are not supported. Moreover, their approach does not provide reasoning about case plan, and the completeness of their proof procedure is unknown.

In the future, we shall further work on proof system for more powerful action logics. We shall consider the implementation of the proposed proof systems on top of **Coq** [10] or **Tableaux** [8], and try to find applications in knowledge representation and reasoning.

## References

- [1] Olderog Ernst-Rüdiger Apt Krzysztof R., de Boer Frank S. *Verification of Sequential and Concurrent Programs*. Springer, third edition, 2009.
- [2] Matteo Baldoni, Laura Giordano, Alberto Martelli, and Viviana Patti. *Reasoning about Complex Actions with Incomplete Knowledge: A Modal*

*Approach*, volume LNCS 2202 of *ICTCS '01*. Springer-Verlag, London, UK, UK, 2001.

- [3] Chitta Baral and Michael Gelfond. Representing concurrent actions in extended logic programming. In *Proceedings of the 13th international joint conference on Artificial intelligence - Volume 2*, pages 866–871, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.
- [4] Chitta Baral, Vladik Kreinovich, and Raúl Trejo. Computational complexity of planning and approximate planning in the presence of incompleteness. *Artificial Intelligence*, 122:241–267, September 2000.
- [5] Marco Cadoli and Francesco M. Donini. A survey on knowledge compilation. *AI Commun.*, 10:137–150, December 1997.
- [6] Oren Etzioni, Steve Hanks, Daniel Weld, Denise Draper, Neal Lesh, and Mike Williamson. An approach to planning with incomplete information. In *In Proc. 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning*, pages 115–125. Morgan Kaufmann, 1992.
- [7] Michael Gelfond and Vladimir Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming*, 17:301–322, 1993.
- [8] Reiner Hähnle. Tableaux and related methods. *Handbook of Automated Reasoning*, 2001.
- [9] G. Neelakantan Kartha. Soundness and completeness theorems for three formalizations of action. *IJCAI93*, pages 724–731, 1993.
- [10] Hector J. Levesque. What is planning in the presence of sensing? In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1139–1146, Portland, Oregon, 1996. American Association for Artificial Intelligence.
- [11] Fangzhen Lin. *Situation Calculus*, chapter 16, pages 649–669. Elsevier, 2007.
- [12] Fangzhen Lin and Ray Reiter. How to progress a database. *Artificial Intelligence*, 92:131–167, 1997.
- [13] Fangzhen Lin and Yoav Shoham. Provably correct theories of action. *J. ACM*, 42:293–320, March 1995.

- [14] Marco De Luca, Enrico Giunchiglia, Massimo Narizzano, and Armando Tacchella. "safe planning" as a qbf evaluation problem. In *Proceedings of the Second RoboCare Workshop*, 2005.
- [15] R. C. Moore. A formal theory of knowledge and action. In J. R. Hobbs and R. C. Moore, editors, *Formal Theories of the Commonsense World*, pages 319–358, Norwood, NJ, 1985. Ablex.
- [16] Davy Van Nieuwenborgh, Thomas Eiter, and Dirk Vermeir. Conditional planning with external functions. *Lecture Notes in Computer Science*, 4483:214–227, 2007.
- [17] Marcelo Oglietti. Understanding planning with incomplete information and sensing. *Artificial Intelligence*, 164(1-2):171–208, May 2005.
- [18] Charles Otwell, Anja Remshagen, and Klaus Truemper. An effective qbf solver for planning problems. In *MSV/AMCS, CSREA Press*, pages 311–316, 2004.
- [19] Ronald P. A. Petrick and Fahiem Bacchus. Extending the knowledge-based approach to planning with incomplete information and sensing. In *In ICAPS-04*, pages 2–11. AAAI Press, 2004.
- [20] Jussi Rintanen. *Planning and SAT*, chapter 15, pages 483–503. Chapter 15, Handbook of Satisfiability, IOS Press, 2009.
- [21] Boaz Barak Sanjeev Arora. Computational complexity:a modern approach. *Cambridge University Press*, 2009.
- [22] Richard B. Scherl and Hector J. Levesque. The frame problem and knowledge-producing actions. In *AAAI*, pages 689–695, 1993.
- [23] Richard B. Scherl and Hector J. Levesque. Knowledge, action, and the frame problem. *Artif. Intell.*, 144:1–39, March 2003.
- [24] Tran C. Son and Chitta Baral. Formalizing sensing actions: A transition function based approach. *Artificial Intelligence*, 125(1-2):19–91, 2001.
- [25] Phan H. Tu, Tran C. Son, and Chitta Baral. Reasoning and planning with sensing actions, incomplete information, and static causal laws using answer set programming. *Theory Pract. Log. Program.*, 7(4):377–450, 2007.